

```

#######
## Program name: pe2b.pl ## 
## Description : Translates hierachical data structures re- ## 
## presented in SGML into relational tables via ## 
## SQL INSERTs statements. ## 
## An Oracle sequence is used to uniquely ## 
## identify every SGML entity at level 2. ## 
## Each SGML element that contains subordinate ## 
## stand-alone elements is associated with a ## 
## table; the columns on each table are the ## 
## stand-alone elements plus all elements in the ## 
## path to the parent element (inclusive). ## 
## Stand-alone repeating elements are not ## 
## supported and therefore are ignored. ## 
## Version      : 0.3, 0.4 ## 
## Contact      : Ricardo Morin (morinr@bah.com) ## 
##                 : Luoming Hou (houl@bah.com) ## 
## Date         : Dec 1996; May 1997 ## 
## Requirements: perl version 5 ## 
##                 nsgmls sofware by James Clark ## 
##                 perlSGML package by Earl Hood ## 
##### 
## Status       : Continue testing ## 
##                 Must remove hard-code reference to DTD file ## 
##                 name and make it a parameter ## 
##### 
## Change Log   : Date      Who      Description ## 
##                 ----- ----- ----- ## 
##                 3/24/97   RM      Add support for embedded single ## 
##                           quotes; embed an additional ## 
##                           single quote as required by SQL ## 
##                 3/25/97   RM      Replace all embedded quotes with## 
##                           two quotes, not just the first ## 
##                           one ## 
##                 4/10/97   RM      Remove embedded newline ## 
##                           characters (it is questionable ## 
##                           whether we should do this or ## 
##                           not) ## 
##                 4/22/97   LH      Replace embeded ";" w/ "'||chr( ## 
##                           95)||'" to prevent SQLPLUS from ## 
##                           blowing up ## 
##                 5/57/97   LH      Replace embeded "\n" w/ "'||chr ## 
##                           (10)||'" to prevent SQLPLUS from## 
##                           blowing up ## 
##                 5/28/97   LH      Add code to generate a PLSQL ## 
##                           stored procedure to include all ## 
##                           INSERT statements w/ EXCEPTION ## 
##                           handles ## 
##### 
## 

require "sgml.pl" || die "Unable to require sgml.pl\n";
require "newgetop.pl" || die "Unable to require newgetop.pl\n";

##### 
## Louis Li found that 'newgetopt.pl' should be 'newgetop.pl' since ## 
## there is no file of 'newgetopt.pl' and there is only file of ## 
## 'newgetop.pl' though in side this file, there is a package of ## 
## 'newgetopt'. 9/19/97: by Luoming Hou and Louis Li. ## 
##### 

##### 
## Globals ## 
##### 
$old = 0;          ## Flag used to ignore <OLD> tags
$level = 0;        ## Points to the "current" level on the hierarchy
$endt = 0;         ## Flag used to detect two consecutive end-tags
$chunk_size = 2000; ## Size of chunkable characters
##### 
## Constants ## 

```

```

#####
$seq_name = "E2B_SEQUENCE"; ## The name of the Oracle sequence
## used to generate IDs

#####
## Data structures
##
#####
#
# lvalues      Hash of stacks containing stand-alone elements
# lnames       Hash of stacks containing names of stand-alone elements
# pvalues      Stack containing path elements
# pnames       Stack containing names of path elements
# exc_elements Lookup hash used to exclude elements from output
#
#
#


#
##


#####
## EndTagFunc
## This is a callback routine from the sgml package.
## It is called every time sgml finds an end tag.
## $ret contains the name of the element.
#####
$sgml::EndTagFunc = sub {
    local($ret) = shift;
    ## Ignore <OLD> tags
    if (uc($ret) eq "OLD") {
        $old = 0
    } else {
        if ($endt ==1) {
            ## Two consecutive end-tags: its time to generate SQL
            output_sql();
            ## Clean-up
            splice(@{$lvalues{$level+1}},0);
            splice(@{$lnames{$level+1}},0);
        }
        ## Clean-up current level
        pop @pvalues;
        pop @pnames;
        $level--;
        $endt = 1;
    }
};

##


#####
## CdataFunc
## This is a callback routine from the sgml package.
## It is called every time sgml finds data.
## $ret contains a reference to the data.
#####
$sgml::CdataFunc = sub {
    ## $/ = ""; RM 4/10/97
    local($ret) = shift;
    ## Ignore <OLD> Tags
    if ($old == 0) {
        if (!(${$ret} =~ /^$/)) {
            ## Remove leading and trailing blanks and newline characters
            ${$ret} =~ s/^[\s*.*?\s*]$/$1/;
            ## chomp ${$ret};
            ## Add single quote for every single quote found (SQL compatibility)
            ${$ret} =~ s/'/\'\''/g; ## RM, 3/24/97

            ## Remove embedded newlines
            ## LH 4/11/97 Replace special characters such as ';' by chr(59):
            ${$ret} =~ s/;/';'chr(59)||'/g; ## LH 4/11/97
            ${$ret} =~ s/\n/chr(10)||'/g; ## LH 5/27/97
        }
    }
};

```

```

## Commented out by LH 5/27/97 for
## ${$ret} =~ s/\n\s+\n/\n/g; ## RM 4/10/97
## ${$ret} =~ s/\n{2,}/\n/g; ## RM 4/10/97

${$lvalues{$level}}[$#{$lvalues{$level}}] .= ${$ret};
$pvalues[$#pvalues] .= ${$ret};

}

};

#
#
#######
##  OpenTagFunc                                ##
##  This is a callback routine from the sgml package.      ##
##  It is called every time sgml finds an end tag.          ##
##  $ret1 contains the name of the tag.                  ##
##  $ret2 contains the attribute part of the element.    ##
######
##$sgml::OpenTagFunc = sub {
local($ret1) = shift;
local($ret2) = shift;
## Ignore <OLD> tags
if (uc($ret1) eq "OLD") {
    $old = 1
}
if ($old == 0) {
    $level++;
    $endt = 0;

    ## Store element name and make placeholder for the data
    push @{$lvalues{$level}}, "";
    push @{$lnames{$level}}, $ret1;
    push @pvalues, "";
    push @pnames, $ret1;

    ## At level two must generate a new ID
    if ($level == 2) {
        $seq_val = "$seq_name.nextval";
    }
}
};

#
#
#######
##  output_sql                                     ##
##  Generates SQL INSERTs for the "current" level      ##
#######
##  Updated by LH, 5/28/97:                         ##
##  Add code to generate exception handle [and concatenations  ##
##  of temp's if necessary] for each INSERT statement w/ value  ##
##  of more than $chunk_size characters                ##
#######
##sub output_sql {
push @out_stack, "\n";

## Code added by LH, 5/28/97:
push @out_stack, " BEGIN \n";
## End of addig code by LH, 5/28/97;

## RM, LH, 5/28/97:
$over_size_flag = 0; ## assume no over size value yet
foreach $i (0 .. $#{$lvalues{$level+1}}) {
    if (!exists $exc_elements{$lnames{$level+1}[$i]}) {

```

```

$ith_val_length = length($lvalues{$level+1}[$i]);
if ($ith_val_length > $chunk_size) {
## There is some value w/ more than $chunk_size of chars,
## and assume that there is at most one of such value:
    $over_size_flag = 1; ## set flag
    $chunks = int($ith_val_length/$chunk_size);
    for ($j=1; $j <= $chunks; $j++) {
        ## generate "    DECLARE" in the begining:
        if ($j eq 1) { ## 1st chunk
            push @out_stack, "    DECLARE\n";
            $concat = "temp1";
        }
        $chunk_text = substr($lvalues{$level+1}[$i],($j-1)*$chunk_size, $chunk_size);
        push @out_stack, "    temp$j VARCHAR2($chunk_size) := \'$chunk_text\';\n";
        if ($j gt 1) { ## not 1st chunk
            $concat = "$concat||temp$j";
        }
    }
    push @out_stack, "    all_temps VARCHAR2($ith_val_length) := $concat;\n";
    push @out_stack, "    BEGIN\n";
}
}

## RM, LH, 5/28/97

push @out_stack, "    INSERT INTO " . $pnames[$#pnames] . " (ID,\n";

## Output names of elements in path first ##
foreach $i (0 .. $#pnames) {
    if ($pvalues[$i] ne "") {
        push @out_stack, "    $pnames[$i],\n";
        ## Exclude last element in path from future output
        $exc_elements{$pnames[$i]} = 1;
    }
}
## Output names of elements at given level
## $delim = "";
foreach $i (0 .. ${$lnames{$level+1}}) {
    if (!exists $exc_elements{$lnames{$level+1}[$i]}) {
        if ($lvalues{$level+1}[$i] ne "") {
            push @out_stack, "$delim    $lnames{$level+1}[$i]";
            $delim = ",\n";
        }
    }
}
push @out_stack, ")\n    VALUES ($seq_val,\n";
## Output values of elements in path first ##
foreach $i (0 .. $#pvalues) {
    if ($pvalues[$i] ne "") {
        push @out_stack, "    \'$pvalues[$i]\',\n";
    }
}
## Output values of elements at given level
## $delim = "";
$discard = 1; ## If all elements at given level are blank, discard
foreach $i (0 .. ${$lvalues{$level+1}}) {
    if (!exists $exc_elements{$lnames{$level+1}[$i]}) {
        if ($lvalues{$level+1}[$i] ne "") {
            $discard = 0;
            ## Add code by LH, 5/29/97:
            if (length($lvalues{$level+1}[$i]) > $chunk_size) {
                push @out_stack, "$delim    all_temps";
            }
        }
    }
}

```

```

        else {
            ## End of adding code by LH, 5/29/97
            push @out_stack, "$delim      \'$lvalues{$level+1}[$i]\'";
        }
    }
}
push @out_stack, " );\n";

## Code added by LH, 5/28/97:
## and updated by LH, 6/16/97:
$table_name = $pnames[$#pnames];
generate_expt_handle();
if ($over_size_flag eq 1) {
    ## generate an extra "END;" to match "BEGIN"
    push @out_stack, " END;\n";
}
## End of adding code by LH, 5/28/97;

if (!$discard) {
    foreach $i (0 .. $#out_stack) {
        print STDOUT ($out_stack[$i]);
    }
    $seq_val = "$seq_name.currval";
}
## Clean-up
splice(@out_stack, 0);
}

#
#
#######
## pop_rep_elements ###
## Reads the DTD to identify repeatable elements so that ###
## they are not treated as stand-alone elements ###
## Hardcoded DTD name (Ich_icsr.dtd) should be generalized ###
#######
###
sub pop_rep_elements {
    open DTD, "Ich_icsr.dtd";
    while (!eof(DTD)) {
        $data = <DTD>;
        ## Find elements that look like <word*> or <word.word*>
        if ($data =~ /(\w+\*)|(\w+\.\w+\*)/) {
            $repv = uc($&);
            chop($repv);
            $exc_elements{$repv} = 1;
        }
    }
    close DTD;
}

#######
## usage ###
## Prints help message ###
###
sub usage {
    print STDERR "$0:\n";
    print STDERR " Translates properly formatted hierachichal SGML data\n";
    print STDERR " into relational tables via SQL INSERTs\n";
    print STDERR "Usage: perl $0 -f filename\n";
    print STDERR " where filename is the name of the SGML\n";
    print STDERR " file to process (assumes SGM file extension)\n";
    print STDERR " Output goes to standard output\n";
    print STDERR " SGML errors go to filename.ERR\n";
    print STDERR " Normalized SGML goes to filename.SGN\n";
    exit(1);
}

```

```

}

#######
## generate_sql_header
## By LH, 5/28/97:
## Generate PLSQL code as the header of a stored procedure
## that includes all the INSERT statements
## Updated
## 6/6/97: Can not user test_io package in PL/SQL!
##          Use dbms_output.put_line instead.
## 6/16/97: Avoid using spool file; Put all errors into a
##          table e2b_load_errors.
#####
## sub generate_sql_header {
##   put plsql stored procedure header into out_stack:
push @out_stack, "WHENEVER SQLERROR EXIT 6;\n";
push @out_stack, "CREATE OR REPLACE PROCEDURE load_e2b AS\n";
push @out_stack, "  ecode VARCHAR2(50);\n";
push @out_stack, "  emess VARCHAR2(300);\n";
push @out_stack, "BEGIN\n";

## Output all code generated:
if (!$discard) {
  foreach $i (0 .. $#out_stack) {
    print STDOUT ($out_stack[$i]);
  }
}
## Clean-up out_stack:
splice(@out_stack, 0);

}

#######
## generate_sql_trailer
## By LH, 5/28/97:
## Generate PLSQL code as the trailer of a stored procedure
## that includes all the INSERT statements
## 7/10/97: add code to insert 'Y' to e2b_commit.done
#####
## sub generate_sql_trailer {
##   put sql trailer into out_stack:
push @out_stack, "\n";

## insert value to table e2b_commit:
push @out_stack, "  insert into e2b_commit values ('Y');\n";

push @out_stack, "  COMMIT;\n";
push @out_stack, "END load_e2b;\n";
push @out_stack, "\n";
push @out_stack, "BEGIN clear_e2b; load_e2b; COMMIT; END;\n";
push @out_stack, "\n";
push @out_stack, "EXIT;\n";

## Output all code generated:
foreach $i (0 .. $#out_stack) {
  print STDOUT ($out_stack[$i]);
}
## Clean-up out_stack:
splice(@out_stack, 0);

}

#######
## generate_expt_handle
## By LH, 5/28/97; Updated 6/6/97:
## Generate exception handle as well as 'END;' part for
##
```

```

## each INSERT statement
#####
sub generate_expt_handle {
    push @out_stack, "      EXCEPTION WHEN OTHERS THEN\n";
    push @out_stack, "          ecode := TO_CHAR(SQLCODE);\n";
    push @out_stack, "          emess := SQLERRM;\n";
    push @out_stack, "          INSERT INTO E2B_LOAD_ERRORS\n";
    push @out_stack, "              (ERR_NUM,PROGRAM,SQL_CODE,SQL_ERRM)\n";
    push @out_stack, "              VALUES\n";
    push @out_stack, "              (E2B_LOAD_ERR_SEQ.NEXTVAL,'load_e2b\\/$table_name',ecode,emess);\n";

    if ($over_size_flag eq 1) {
        ## Align 'END' w/ 'EXCEPTION':
        push @out_stack, "      END;\n";
    }
    else {
        ## Align 'END' w/ 'BEGIN'
        push @out_stack, "      END;\n";
    }
}

#
#
#######
##  Main
#######
##  Updated by LH, 5/28/97:
##  1. Generate PLSQL stored procedure header in begining;  ##
##  2. Generate PLSQL stored procedure trailer in the end;  ##
##  This stored procedure includes all INSERT statements.  ##
#######

## Get and validate input parameters
&usage() unless &NGetOpt("f=s");

if (!defined $opt_f) {
    print STDERR "$0: Must enter name of file to process\n";
    &usage();
} elsif ($opt_f =~ /\./) {
    print STDERR "$0: Sorry, no extensions\n";
    &usage();
} elsif (!(-r "$opt_f.sgm")) {
    print STDERR "$0: Cannot access $opt_f.sgm\n";
    &usage();
}

## Populate list of
pop_rep_elements();

## Run the SGML parser/normalizer
system "sgmlnorm -f$opt_f.err Ich_icsr.dtd $opt_f.sgm >$opt_f.sgn";

## Generate PLSQL stored procedure header:
generate_sql_header();

## Traverse the SGML document (this is what does it!)
open NSGML, "$opt_f.sgn";
&SGMLread_sgml(\*NSGML);

## Generate PLSQL stored procedure trailer:
generate_sql_trailer();

```